KIPS Trans. Comp. and Comm. Sys.
Vol.10, No.5 pp.137~144
ISSN: 2287-5891 (Print), ISSN 2734-049X (Online)

https://doi.org/10.3745/KTCCS.2021.10.5.137

Performance Analysis of QUIC Protocol for Web and Streaming Services

Hye-Been Nam† · Joong-Hwa Jung† · Dong-Kyu Choi† · Seok-Joo Koh††

ABSTRACT

The IETF has recently been standardizing the QUIC protocol for HTTP/3 services. It is noted that HTTP/3 uses QUIC as the underlying protocol, whereas HTTP/1.1 and HTTP/2 are based on TCP. Differently from TCP, the QUIC uses 0-RTT or 1-RTT transmissions to reduce the connection establishment delays of TCP and SCTP. Moreover, to solve the head-of-line blocking problem, QUIC uses the multi-streaming feature. In addition, QUIC provides various features, including the connection migration, and it is available at the Chrome browser. In this paper, we analyze the performance of QUIC for HTTP-based web and streaming services by comparing with the existing TCP and Streaming Control Transmission Protocol (SCTP) in the network environments with different link delays and packet error rates. From the experimental results, we can see that QUIC provides better throughputs than TCP and SCTP, and the gaps of performances get larger, as the link delays and packet error rates increase.

Keywords : QUIC, SCTP, TCP, Performance Analysis, Web, Streaming

웹 및 스트리밍 서비스에 대한 QUIC 프로토콜 성능 분석

남 혜 빈† · 정 중 화† · 최 동 규† · 고 석 주††

요 약

최근 IETF에서는 HTTP/3.0 기반 웹서비스 제공을 위하여 QUIC 프로토콜 표준을 개발중에 있다. 기존 HTTP/1.1 및 HTTP/2에서는 수송계층 프로토콜로서 TCP를 사용하는 데 비하여, HTTP/3에서는 QUIC/UDP를 사용한다. TCP와는 달리 QUIC에서는 연결설정에 소요되는 시간을 단축시키기 위해 0-RTT 혹은 1-RTT 기법을 사용하고, TCP의 head-of-line blocking 문제를 해결하기 위해 멀티스트리밍(multi-streaming) 기법을 사용한다. 이 밖에도 connection migration 등 다양한 특징을 제공하고 있으며, 구글의 크롬 브라우저에서 시험용 코드를 제공하고 있다. 이 논문에서는 HTTP 기반의 웹 서비스 및 스트리밍 서비스를 대상으로 QUIC 프로토콜에 대한 성능 분석을 수행한다. 실제 오픈 소스 커뮤니티에서 제공하는 코드를 활용하여 소규모 테스트베드를 구성하고, 다양한 링크 지연시간, 패킷오류율을 갖는 네트워크 환경에서 TCP 및 SCTP(Streaming Control Transmission Protocol) 프로토콜과 QUIC 프로토콜의 성능을 비교하였다. 실험 결과, QUIC은 TCP 및 SCTP에 비해 더 좋은 성능을 보이며, 특히 링크 지연시간 및 패킷오류율이 높을수록 기존 프로토콜과의 성능 격차는 더욱 커짐을 확인하였다.

키워드 : QUIC, SCTP, TCP, 성능 분석, 웹, 스트리밍

1. 서 론

최근 다양한 사물인터넷 및 모바일 인터넷 서비스가 급격히 증가함에 따라 인터넷 트래픽을 더욱 빠르고 효율적으로 전송하기 위한 기술 개발의 중요성이 커지고 있다. 특히, HTTP 기반의 웹 서비스의 성능 개선을 위해 많은 연구들이 진행되었으며, 주로 TCP, Stream Control Transmission Protocol(SCTP), UDP [1-3] 등의 기존 수송 계층 프로토콜을 효율적으로 활용하는 이슈들이 논의되어 왔다[4,5].

한편, 다양한 웹 기반 서비스가 등장함에 따라 HTTP 프로토콜도 더욱 다양한 기능과 높은 성능을 제공하기 위해 꾸준히 발전하여 2015년도에 HTTP/2가 표준화되었다. HTTP/2는 HTTP/1.1에 비해 체감 지연 시간을 현저히 개선하였다. 하지만 HTTP/2 또한 TCP 기반으로 설계되었기 때문에 head-of-line blocking(HoLB) 문제 및 느린 연결설정 시간 등과 같은 TCP의 비효율적 특징을 모두 상속하였다. 한편, 인터넷 표준화 기구인 Internet Engineering Task Force(IETF)에서는 이러한 문제점을 개선하기 위해 QUIC 프로토콜 표준을 개발 중에 있다 [6-9]. 참고로 QUIC은 개발 초기에 Quick UDP Internet Connection의 약자로 사용되었지만 지금은 QUIC 단어 자체가 고유 명사화되었다 [10].

※ 이 연구는 2021년도 산업통상자원부 및 산업기술평가관리원(KEIT) 지원에 의해 수행되었음(20002214).
† 비 회 원 : 경북대학교 컴퓨터학부 박사과정
†† 종신회원 : 경북대학교 컴퓨터학부 교수
Manuscript Received : September 17, 2020
Accepted : October 22, 2020
* Corresponding Author : Seok-Joo Koh(sjkoh@knu.ac.kr)

Table 1. Comparison of Transport Protocols

Category	TCP	SCTP	QUIC
Connection Identifier	IP, Port	IP, Port	Connection ID
ACK mechanism	Cumulative ACK	Selective ACK	Selective ACK
Multi-streaming	Not Support	Support	Support
Multi-homing	Not Support	Support	Support
Connection setup process	three-way	four-way	0-RTT, 1-RTT

TCP는 신뢰성을 제공하는 전통적인 수송계층 프로토콜로서 흐름 제어, 혼잡 제어, 오류 제어 등의 다양한 기능을 제공한다. 하지만 초지연 실시간 전송을 요구하는 인터넷 서비스의 증가로 인해 TCP의 한계점이 드러나고 있다.

SCTP는 TCP의 단점을 개선하고 인터넷 전화 등의 실시간 고품질 서비스 전송을 위해 개발된 수송계층 프로토콜이다. 특히, SCTP는 HoLB 문제를 극복하기 위한 멀티스트리밍(multi-streaming) 및 여러 개의 IP 주소를 사용하는 멀티홈잉(multi-homing) 특징을 제공한다.

QUIC은 Internet Engineering Task Force(IETF)에서 표준으로 제정 중인 UDP 기반의 수송계층 프로토콜로서 기존 수송 계층 프로토콜들의 한계를 극복하기 위해 개발되었다. 현재 구글(Google) 주도로 표준개발이 진행되고 있으며, 차세대 웹 전송 프로콜인 HTTP/3부터는 하위 계층으로 QUIC 프로토콜을 사용할 예정이다.

Table 1에서는 지금까지 기술한 TCP, SCTP 및 QUIC 프로토콜들의 특징을 비교하고 있다. 모두 신뢰전송 기능을 제공하지만 서로 다른 특징을 가진다. QUIC의 경우 SCTP와 유사한 특징을 제공하지만, 연결 식별자(connection ID)를 사용하고 0-RTT 및 1-RTT 전송 기능을 사용하여 초기 연결 설정 시간을 대폭 줄이는 특징을 제공한다.

한편, 최근에 QUIC 성능분석 관련 논문이 일부 제시되고 있으나 대부분의 논문들이 간단한 웹 서비스 환경에서의 성능 분석을 다루고 있다. 하지만 차세대 수송계층 프로토콜로서 QUIC의 확장가능성을 고려할 때에, 스트리밍 서비스 등 다양한 특징을 갖는 트래픽에 대한 성능 분석이 요구된다. 따라서, 이 논문에서는 웹 및 스트리밍 서비스에 대하여 다양한 네트워크 환경에서 QUIC 프로토콜의 성능을 비교 분석하고자 한다.

이 논문의 구성은 다음과 같다. 먼저 2절에서는 기존 TCP, SCTP 및 QUIC 프로토콜의 특징들을 비교한다. 3절에서는 HTTP/2 및 HTTP/3 기반의 웹 서비스에 대하여 TCP 및 QUIC 프로토콜의 성능을 비교 분석한다. 4절에서는 스트리밍 서비스에 대하여 TCP, SCTP 및 QUIC 프토콜의 성능을 다양한 네트워크 환경에서 비교 분석한다. 끝으로 5절에서 이 논문의 결론을 맺는다.

2. 수송 계층 프로토콜의 특징 비교

2.1 TCP

TCP에서 서버와 클라이언트는 3단계(three-way handshake) 과정을 통해 연결을 설정하고 데이터 전송을 시작한다. 설정된 연결 정보를 토대로 흐름 제어, 혼잡 제어 및 오류 제어 기능을 수행한다[11,12].

각 송신 패킷은 일련번호(sequence number)를 포함하고 수신자는 승인번호(acknowledgement number)를 통해 송신자에게 패킷의 수신 상태 정보를 제공한다. 이때 전달되는 승인번호는 누적(cumulative) 성격을 갖는다.

2.2 SCTP

TCP는 HoLB 문제 등의 구조적인 문제로 인해 효율적인 인터넷 데이터 전송에 한계를 갖는다. IETF에서는 2000년경에 인터넷 전화 등의 실시간 신뢰 전송 서비스를 위하여 SCTP를 표준으로 제정하였다.

SCTP는 TCP처럼 연결지향 프로토콜이다. SCTP 연결(association)은 여러 개의 IP 주소가 연계될 수 있다. TCP와 달리 SCTP 연결설정은 4단계 절차를 사용한다. 즉, 연결설정 과정에서 송수신자간에 쿠키(cookie) 정보를 교환하여 Denial of Service(DoS) 공격을 방지하기 위해서 한 단계가 추가되었다.

또한, SCTP는 ACK 메커니즘으로서 Cumulative TSN 값과 함께 Gap ACK 블록을 함께 전송하는 Selective ACK [13] 기법을 사용한다. 패킷 구조 및 프로토콜 동작 측면에서, 바이트 기반의 TCP와는 달리 SCTP는 메시지 기반 방식을 사용한다. SCTP 패킷은 기본 헤더와 함께 여러 개의 제어 청크(chunk) 및 데이터 청크로 구성된다.

TCP와 차별되는 SCTP의 고유 특징은 멀티스트리밍 및 멀티홈잉이다. SCTP 멀티스트리밍 기능을 통해 하나의 SCTP 연결은 여러 개의 데이터 스트림을 전송할 수 있으며, 이를 통해 TCP의 HoLB 문제를 완화시킬 수 있다. 이를 위해 각 데이터 청크는 각 스트림의 구분을 위한 Stream ID 정보와 스트림별 Stream Sequence Number(SSN) 정보를 포함하여 데이터 전송 관리에 활용한다. 즉, 특정 스트림에서 패킷 오류가 발생하더라도, 다른 스트림 데이터들은 이에 영향을 받지 않고 수신자 응용에 전달될 수 있는 장점이 있다.

한편, SCTP 멀티홈잉 기능을 통해 여러 개의 IP 주소가 SCTP 연결에 사용될 수 있다. 여러 개의 IP 주소 중에서 하나의 주소가 primary 주소로 활용되어 통상적인 데이터 전송에 활용되며, 나머지 IP 주소들은 backup IP 주소로 활용된다. 즉, 네트워크 장애 등으로 primary IP 주소를 사용할 수 없는 경우에는 즉각적으로 backup IP 주소가 primary IP 주소로 변경되어 사용된다. TCP의 경우, 네트워크 장애가 발생하면 TCP 연결설정을 다시 시작해야 하므로 프로토콜 성능에 심각한 저하를 가져오는 반면에, SCTP는 멀티홈잉 기능을 통해 전송 성능을 유지할 수 있다.

SCTP는 위와 같은 기술적 효율성에도 불구하고 실제 인터넷 응용 서비스에서는 잘 활용되고 있지 않다. 주된 이유로는 네트워크 고착화(ossification) 문제로 인하여, 패킷 라우팅 및 트래픽 감시 등을 위해 네트워크에 설치된 다양한 미들박스(middle box)에서 SCTP 패킷 전달 기능을 지원하지 않기 때문이다.

2.3 QUIC

인터넷이 개발된 이후 매우 다양한 서비스들이 개발되었으며, 그 중 대부분은 HTTP 기반 웹 서비스이다. 구글에서는 2015년에 웹 접속 속도를 개선하기 위해 'SPDY'라는 자체 프로토콜을 개발하였다. SPDY 프로토콜은 HTTP 헤더 압축, 바이너리 프로토콜, 멀티플렉싱 등의 특징을 제공한다. 이는 웹 서비스의 전송 지연시간 단축에 유용하게 사용되었으며 HTTP/2 표준의 근간이 되었다. 하지만, HTTP/2은 하위계층 프로토콜로서 여전히 TCP를 사용한다. 따라서 TCP의 문제점인 긴 연결 설정 시간 문제 및 HoLB 문제 등을 안고 있었다.

기존 수송 계층 프로토콜 문제의 근본적인 해결을 원했던 IETF는 2013년부터 UDP 기반 수송 프로토콜인 QUIC 표준 개발을 시작하였다. QUIC은 수송 계층 프로토콜이지만 UDP 위에서 동작하도록 설계되어 SCTP와는 달리 배포 및 보급이 용이하다.

QUIC 프로토콜은 앞서 언급한 SPDY 기능과 함께 TLS (Transport Layer Security) 1.3 보안 기능을 포함하여 설계되었다. 이처럼 QUIC은 다양한 특징을 제공하고 있으며 이 논문에서는 이 중에서 연결 설정 관리, 멀티스트리밍 기능 및 오류 제어 기능을 살펴본다.

1) 연결 설정: 0-RTT 및 1-RTT

TCP와 SCTP처럼 QUIC은 응용에게서 전달받은 데이터 교환을 위해 클라이언트-서버간 연결설정을 수행한다[14]. QUIC 서버와 클라이언트는 연결설정 과정을 통해 데이터 전송 및 암호화에 필요한 정보를 교환한다.

QUIC의 가장 큰 특징 중 하나는 TLS 1.3을 내부적으로 포함했다는 점이다. 인증, 암호화, 무결성을 제공하는 TLS는 인터넷 서비스에서 필수적인 프로토콜이다. TCP와 TLS는 완전 독립적인 프로토콜이기 때문에 TLS로 암호화된 통신을 수행할 경우 TCP의 연결 과정 이후 암호화 방식 협상 및 암호화 키를 공유하기 위한 TLS의 연결 과정을 별도로 거쳐야 한다. 하지만 QUIC의 경우 프로토콜 내에 TLS를 포함하여 메시지 전달에 필요한 인수 교환 및 암호화에 필요한 절차가 동시에 수행된다.

Fig. 1은 TCP+TLS1.3 및 QUIC+TLS1.3[15,16] 방식에서의 연결설정 과정을 보여준다. Fig. 1(a)은 클라이언트가 서버에 처음으로(first connection) 접속하는 경우를 의미한다. 이 경우 TCP+TLS1.3 방식에서는 TCP 연결설정과 TLS 설정을 위해 최소 2-RTT(Round Trip Time)가 요구된다. 반면에 QUIC+TLS1.3에서는 TLS1.3 기능이 QUIC 내부에 포함되어 있으므로 1-RTT 후에 데이터 전송이 가능하다. 한편, Fig. 1(b)는 두 번째 연결 (subsequent connection) 혹은 이미

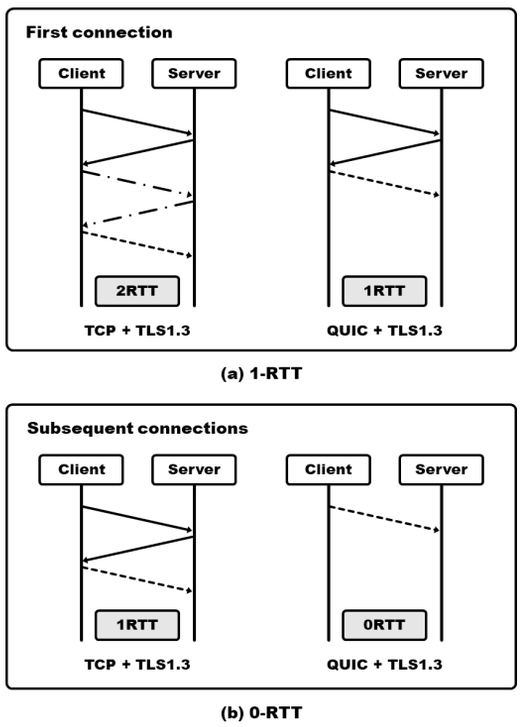

Fig. 1. Connection Establishment of TCP and QUIC

서버에 클라이언트 정보가 저장되어 있는 경우를 의미한다. 이 경우, QUIC+TLS1.3방식에서 클라이언트는 0-RTT 즉, 연결 설정 과정 없이 곧 바로 데이터를 서버에게 전송할 수 있다. 반면에 TCP 방식에는 TCP 연결설정 과정을 위해 1-RTT가 소요된다.

이처럼 TLS를 포함한 QUIC에서는 메시지 교환 및 암호화, 인증에 필요한 정보를 한 번에 교환할 수 있어 효율적인 연결 설정 절차를 가진다는 것을 알 수 있다. 또한, QUIC 에서 클라이언트는 첫 번째 연결 중에 협상 된 인수, 암호화 키를 계산하는데 사용되는 서버의 Diffie-Hellman 값을 저장해놓고, 후속 연결에서 데이터 패킷과 함께 이 값들을 서버로 전송하기 때문에 0-RTT 연결이 가능하다. 이러한 특징은 웹 서비스에서 페이지 로딩 시간을 줄이는 데에 효과적으로 사용될 수 있다.

2) 멀티스트리밍(multi-streaming) 전송

SCTP와 마찬가지로 QUIC에서 하나의 연결은 다수의 데이터 스트림으로 구성될 수 있다. 스트림은 단방향 스트림과 양방향 스트림 등 2개의 타입으로 정의될 수 있다.

QUIC 패킷은 long header 혹은 short header 타입의 기본 헤더를 가진다. long header의 경우 연결 설정, 버전 협상과 같이 특수한 경우에만 사용되며, 연결 설정 이후 나머지 패킷들은 short header를 사용한다.

Fig. 2는 데이터 전달에 사용되는 QUIC의 short header와 스트림(STREAM) 프레임의 형태를 보여준다. STREAM 프레임을 보면 기본 헤더에 있는 Connection ID와 별개로 Stream ID가 있음을 확인할 수 있다. QUIC은 한 연결 내에 다중 스트림을 제공하고, 개별 스트림 내에서 흐름을 제어할 수 있도록 함

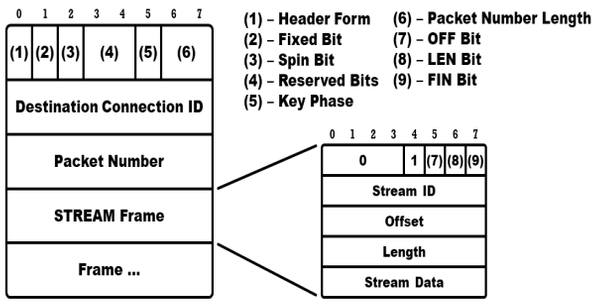

Fig. 2. Short Header and STREAM Frame of QUIC

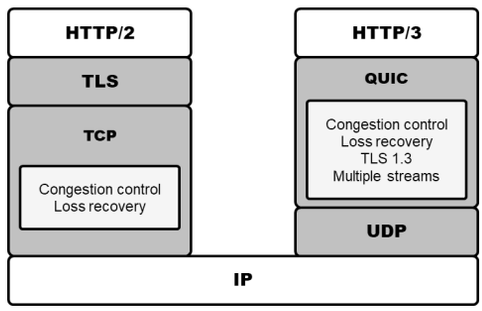

Fig. 3. Protocol Stacks of TCP and QUIC for HTTP

으로써 TCP의 HoLB 문제를 효율적으로 해결할 수 있다.

3) SACK 기반 오류 제어

QUIC은 TCP 및 SCTP처럼 오류제어 기능을 제공한다. QUIC은 오류가 발생하면 재전송을 통해 에러를 복구하는 Automatic Repeat Request(ARQ) 방식과 수신측에서 에러 검출 및 수정을 수행하는 Foward Error Correction (FEC) 방식을 모두 고려했었으나, FEC의 효율성이 입증되지 못하여 현재 표준에는 ARQ 방식만 사용하고 있다.

한편, QUIC은 오류 탐지를 위해 Selective ACK(SACK) 방식을 사용한다. SCTP처럼 QUIC 또한 기본 헤더가 아닌 별도의 ACK 프레임을 통해 ACK 매커니즘을 제공한다. ACK 프레임은 하나의 Largest Acknowledged와 다수의 ACK Range 필드로 구성된다. Largest Acknowledged는 수신한 패킷의 packet number 중 가장 큰 값이 설정된다. ACK Range 필드는 다시 Gap과 ACK Range Length 필드로 나뉘는데 이 두 값을 이용해 정상적으로 수신한 packet number를 모두 표현한다.

4) Connection ID 및 Connection Migration

이 외에도 QUIC는 다양한 특징들을 제공한다. QUIC 연결은 4-tuple(Source IP, Source Port, Destination IP, Destination Port)이 아닌 'Connection ID'로 식별한다. Connection ID를 사용하면 IP 주소 혹은 포트 번호가 변경되는 경우에도 지속적으로 연결을 유지할 수 있다.

예를 들어 이동통신 환경에서 QUIC을 사용하여 파일을 다운로드 중인 클라이언트 단말이 WiFi가 지원되는 곳으로 이동하는 경우에도, 끊김 없이 WiFi 네트워크를 통해 남은 파일을 전달받을 수 있다. 이러한 기능을 'connection migration'이라 하며, 앞에서 언급한 1-RTT, 0-RTT 연결 설정과 함께 QUIC의 대표적인 특징이다.

connection migration 기능은 또한 NAT(Network Address Translator)를 사용하는 상용 인터넷망에서도 유용하게 사용될 수 있다. 즉, NAT 장비에서 클라이언트의 IP주소와 포트번호가 주기적으로 변경될 수 있는데, 이러한 경우에도 클라이언트는 Connection ID 기능을 활용하여 서버와 지속적인 연결을 유지할 수 있다.

3. HTTP 기반 웹 서비스 성능 비교 분석

3.1 개요

이 절에서는 HTTP 기반 웹 서비스에 대하여 TCP와 QUIC 프로토콜의 성능을 비교 분석한다. Fig. 3은 성능 비교 실험을 위해 사용한 프로토콜 스택 구조를 보여준다. 먼저, TCP를 활용하여 HTTP/2 데이터를 전송하는 실험을 수행하고, 아울러 QUIC을 사용하여 HTTP/3 데이터를 전송하는 실험을 수행한다.

QUIC은 C++, Golang, RUST, Python 등 다양한 프로그래밍 언어로 구현된 라이브러리 및 프레임워크가 존재하며 [17-19], 이미 많은 서버들이 QUIC을 통해 웹 서비스를 제공하고 있다. 특히, Google Chrome과 Mozila Firefox 브라우저에서 시험적으로 QUIC 프로토콜을 지원하고 있다. 구글 보고서에 의하면 QUIC 프로토콜을 자사의 서비스에 적용한 결과, 약 3%의 웹 페이지 로딩 시간을 개선하였으며, 유튜브 스트리밍 서비스의 경우 30% 정도의 버퍼링이 감소하였음을 발표하였다.

3.2 실험 환경 구성

이 논문에서는 HTTP/2와 HTTP/3의 Page Loading Time(PLT)을 비교 분석하는 실험을 위해, Fig. 4와 같이 간단한 테스트베드를 구축하였다.

실험에서 서버와 클라이언트는 각각 Wireless Access Point(AP)에 연결되어 있으며, 서버와 클라이언트 사이의 지연시간은 200ms로 설정하였다. 오픈 소스 웹 프레임워크인 Caddy[20]를 서버 프로그램으로 사용하였으며, HTML, CSS, JS 파일과 5개의 이미지로 구성된 웹 페이지를 구성하였다. 또한, Chrome 브라우저와 'quic-go' 라이브러리를 기반으

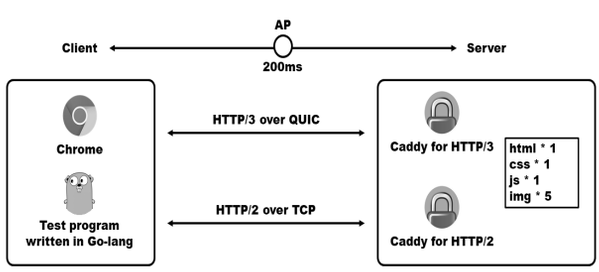

Fig. 4. Testbed Configuration for PLT Analysis

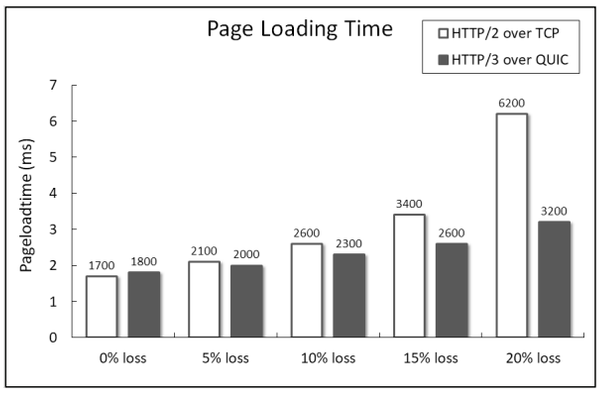

Fig. 5. Page Loading Times for Different Error Rates

로 직접 구현한 프로그램을 클라이언트로 사용하였으며, 클라이언트에서 서버에 접속하여 웹 문서를 다운로드 하여 재생하는 데 소요되는 시간을 측정하였다.

3.3 PLT 성능 분석

QUIC 성능 분석 관련 기존 논문들은 연결 관리, 멀티스트리밍과 같은 QUIC의 특징들이 지연이나 오류율이 높은 환경 또는 핸드오버가 많은 환경에서 웹 서비스의 성능을 비교하였다[21]. 이 논문에서는 평균 200ms의 지연을 가지는 네트워크 환경에서 패킷오류율(packet error rate)을 20%까지 증가시키면서 HTTP/2 over TCP 및 HTTP/3 over QUIC의 PLT를 측정하여 비교하였다.

Fig. 5는 다양한 패킷오류율에 대한 HTTP/2 over TCP와 HTTP/3 over QUIC 방식의 PLT를 비교한 결과이다. 그림에서 오류가 없는 환경에서는 두 가지 방식의 성능이 비슷하지만, 오류율이 높은 환경에서는 QUIC이 더 좋은 성능을 보여줌을 확인할 수 있다. 특히 에러율이 20%일 경우 QUIC 방식은 TCP 방식에 비하여 2배 정도의 성능 차이를 보이고 있다. 이러한 성능차이는 QUIC 프로토콜의 0/1-RTT 연결 설정 기능, SACK 기반 오류제어 및 멀티스트리밍 특성에 기인한 것으로 분석된다.

4. 스트리밍 서비스 기반 성능 분석

4.1 실험 환경 구성

이 절에서는 멀티미디어 스트리밍 트래픽을 대상으로 TCP, SCTP 및 QUIC의 성능을 다양한 네트워크 환경에서 비교 분석한다. 먼저 미디어 전송에 대한 성능 분석을 위해 간단한 테스트베드를 구현하였다.

Fig. 6은 테스트베드 구성을 보여준다. 테스트베드는 서버, 클라이언트, 네트워크 환경 매니저 및 2개의 AP로 구성되어 있다. 서버는 미디어 파일을 전송하며, 클라이언트는 재생할 미디어 파일을 요청하고, 전달받은 미디어 파일을 재생한다. 네트워크 환경 매니저를 통해 링크지연(link delay), 오류율(error rate) 및 재연결 요청(reconnection request) 등의 실험 환경을 변경하였다. 서버는 데스크톱 PC를, 클라

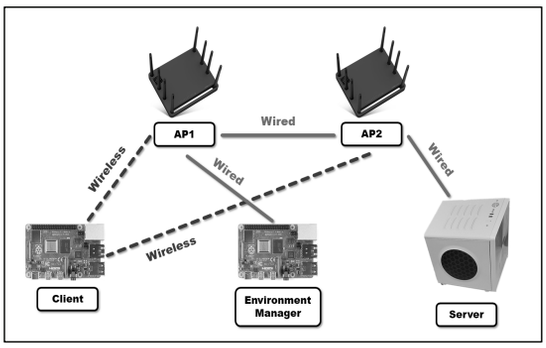

Fig. 6. Testbed Configuration for Performance Analysis

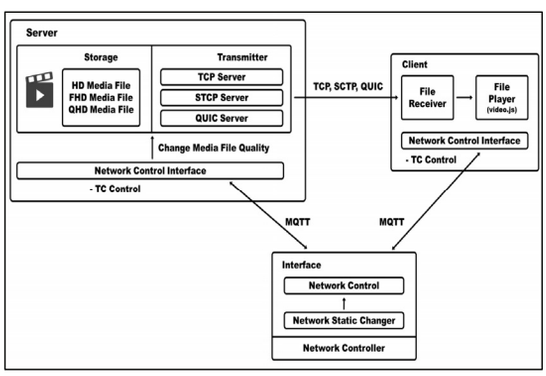

Fig 7. Testbed Logical Module

이언트와 실험환경 매니저는 라즈베리 파이를 사용하였다. 네트워크 환경 매니저는 AP1에 연결하였으며, 서버는 AP2에 연결하였다. 클라이언트는 네트워크 환경 매니저의 제어에 따라 AP1 또는 AP2에 무선으로 연결된다.

Fig. 7은 서버, 클라이언트, 네트워크 환경 매니저를 구성하는 논리적 모듈을 보여준다. 본 실험에서 사용된 소프트웨어는 Go 언어를 기반으로 구현되었으며, TCP는 내장된 net 패키지를, SCTP와 QUIC은 오픈 소스로 구현된 sctp 패키지와 quic-go 패키지를 사용하였다[22,23].

네트워크 환경 매니저는 네트워크 상태 변화기(Network Status Changer, NSC), Seek 변화기(Seek Changer)와 실험환경 제어 인터페이스로 구성되어 있다. 네트워크 상태 변화기는 특정 주기로 링크 지연과 재연결 여부를, Seek 변화기는 Seek 이동을 위한 값을 무작위로 추출한다. 추출된 값은 실험환경 제어 인터페이스로 서버와 클라이언트에 전달한다.

서버는 미디어 파일 요청을 처리하는 노드로 저장소(Storage), 전달 모듈 (Transmitter Module), 환경변수 제어 인터페이스(Environment Control Interface)로 구성되어 있다. 저장소에는 High-Definition(HD), Full High-Definition(FHD), Quad High-Definition(QHD) 해상도를 가지는 미디어 파일들이 저장되어 있다. 모든 미디어 파일들은 약 1초의 런타임을 가진다. 전달 모듈은 클라이언트로부터 요청받은 미디어 파일을 전달하는 간단한 TCP, SCTP, QUIC 서버로 구성되어 있다. 환경변수 제어 인터페이스는 실험환경 매니저로부터 환경변수를 수신하여 실제 환경에 적용하는 역할을 한다.

본 실험에서 링크 지연과 오류율은 Traffic Control(TC) 프로그램을 통해 제어하였다.

클라이언트는 재생할 미디어 파일을 요청하고 수신한 파일을 재생하는 노드로서, 수신 모듈(receiving module), 미디어 재생기(media player), 실험환경 제어 인터페이스로 구성되어 있다.

실험 순서는 다음과 같이 진행되었다. 먼저 수신 모듈은 재생할 미디어 파일을 서버에 요청한다. 이때 링크 지연이 100ms 보다 작다면 QHD 미디어 파일을, 100ms~200ms 라면 FHD 미디어 파일을, 200ms 보다 크다면 HD 미디어 파일을 요청한다. 연결 직후 또는 Seek가 변경된 후에는 반드시 HD 미디어 파일을 요청한다. 서버와 마찬가지로 클라이언트 또한 네트워크 환경 매니저로부터 환경변수를 수신하여 실제 환경에 적용한다.

4.2 성능 분석 결과

성능 척도로서 Total Transmission Delay(TTD)를 측정하였다. TTD는 클라이언트가 서버에 서비스를 요청하고, 서버로부터 파일을 받기까지 소요되는 전체 시간을 의미한다. TTD는 다양한 네트워크 환경에서 프로토콜의 처리율(throughput)과 관련된 지표이다. 우리는 TTD를 측정하기 위해 1GB 사이즈의 파일 전송 시간을 측정하였으며, 5번의 실험 후 측정된 평균 전송 시간의 평균을 구하였다. 또한, 다양한 네트워크 환경을 실험하기 위해서 평균 링크 지연시간(Average Link Delay, ALD)을 변경해 가며 각 프로토콜별로 TTD를 측정하였다. 각 실험에서 NSC는 주어진 ALD를 평균값으로 하는 정규분포 함수를 이용하여 Link Delay 값을 결정한다.

Fig. 8은 다양한 ALD 지연값에 따른 TTD 성능 비교 결과를 보여준다. 그림에서 멀티스트리밍을 지원하는 SCTP와 QUIC 방식이 TCP 방식 보다 전반적으로 더 좋은 TTD 성능을 보여줌을 확인할 수 있다. 이는 Link Delay가 변경되었을 때 멀티스트리밍 기능을 사용하여 상황에 맞는 파일 전송을 수행할 수 있기 때문이다. 응용 계층에서 개발된 QUIC은 응용으로부터 수신한 데이터, 패킷 번호, 최대 전송 단위(MTU), TLS 키 정보 등을 한꺼번에 처리하여 효율적인 통신을 지원한다. 또한, QUIC은 SCTP와 달리 데이터 전송 중에 필요에 따라 스트림의 수를 유연하게 늘릴 수 있어 지연 상황에 따라 기존 스트림이 아닌 새로운 스트림으로 데이터를 전송함으로써 더욱 빠르게 상황에 적응할 수 있었다. 이러한 특성과 함께 0/1-RTT 특성을 제공하여 높은 지연 환경에서 SCTP 보다 나은 성능을 본 실험에서 보여주었다. Link Delay가 클수록 후보 프로토콜간 성능 차이도 조금씩 더 커지고 있음을 확인할 수 있다.

Fig. 9는 다양한 패킷오류율(error rate)에 대한 프로토콜의 성능을 비교한 결과이다. 그림에서 알 수 있듯이 패킷오류가 없는 환경에서는 프로토콜간 성능 차이가 크지 않음을 알 수 있다. 하지만 패킷오류율이 증가할수록 SACK 기반 오류제어 및 멀티스트리밍을 지원하는 SCTP와 QUIC 방식이 좋은 성능을 보여주었다. 또한, QUIC은 바이트 스트림 추상화 및 우선순위 기반 재전송 등 효율적인 에러 처리 기법을 제공하며,

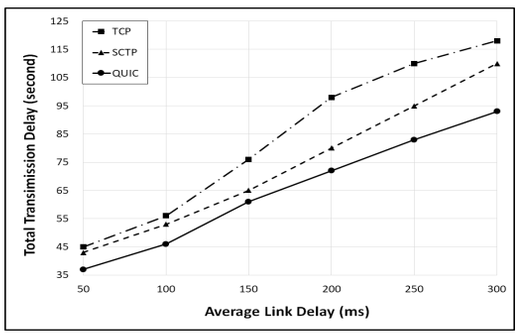

Fig. 8. TTD for Different Average Link Delay

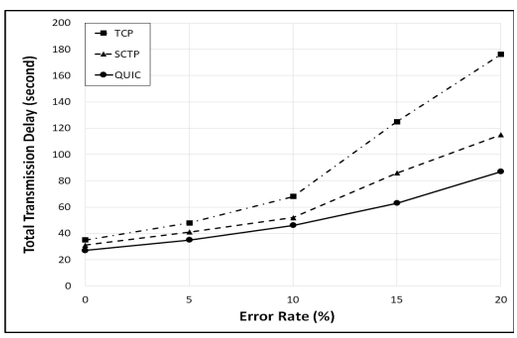

Fig. 9. TTD for Different Error Rate

0/1-RTT 연결설정으로 연결설정 시간을 획기적으로 줄였다. QUIC은 이러한 특성으로 패킷 오류율이 증가하는 상황에서 가장 좋은 성능을 보여주었다. 그림에서 패킷오류율이 높을수록 프로토콜 성능 차이가 더 커짐을 확인할 수 있다.

Fig. 10은 재연결 횟수에 따른 프로토콜의 성능 비교 결과이다. 실험에서 스트리밍 전송 동안에 인위적으로 재연결 과정으로 수행하도록 설정하였다. 이는 QUIC 프로토콜의 0/1-RTT 기능에서 대한 성능분석을 수행하기 위함이다. 그림에서 알 수 있듯이, QUIC 프로토콜이 가장 좋은 성능을 보여주고 있다. 특히, 재연결 횟수가 증가할수록 프로토콜 간 성능 차이는 더욱 커지고 있음을 알 수 있디.

한편, 최근 스트리밍 서비스들은 사용자들에게 향상된 Quality of Experience(QoE)를 제공하기 위해서, 서비스에 대한 전체 리소스가 아닌 당장 필요한 주요 리소스를 먼저 요청하여 재생함으로 반응시간을 줄이는 기법들이 요구된다. QoE에 대한 성능분석을 위해 Fig. 11, 12, 13에서는 로딩 지연 시간(Loading Delay)을 비교하고 있다. Loading Delay는 클라이언트가 서버에 순간 실행할 미디어 파일을 요청한 후 수신할 때까지 소요된 대기 시간 및 전송 시간의 합으로 측정하였다. LD를 구하기 위해 사용된 미디어 파일은 전체 미디어 파일을 잘게 나눈 파일로 약 4~5 MB의 사이즈를 가진다. 또한, 사용자가 재생할 시간을 옮기는 상황을 시뮬레이션하기 위해 매초마다 30%의 확률로 Seek의 위치를 무작위로 움직였다.

Fig. 11은 다양한 링크 지연 환경에서 세 가지 프로토콜의 Loading Delay를 보여준다. 그림에서 알 수 있듯이 링크 지연이 낮은 환경에서는 TCP에 비하여 SCTP와 QUIC 방식이 좋은 성능을 보여주지 못했다. 하지만 다른 해상도를

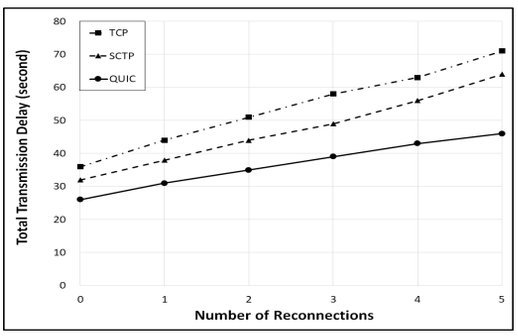

Fig. 10. TTD for Different Number or Reconnections

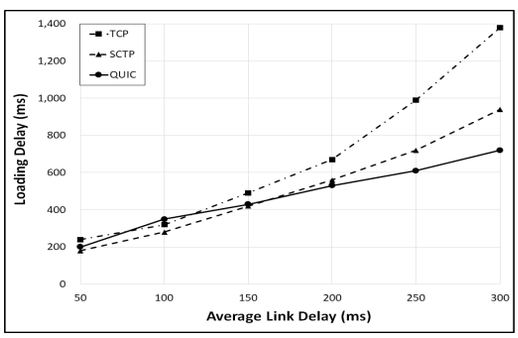

Fig. 11. Loading Delay for Different Average Link Delays

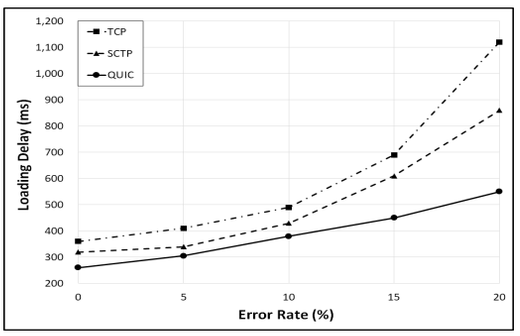

Fig. 12. Loading Delay for Different Error Rates

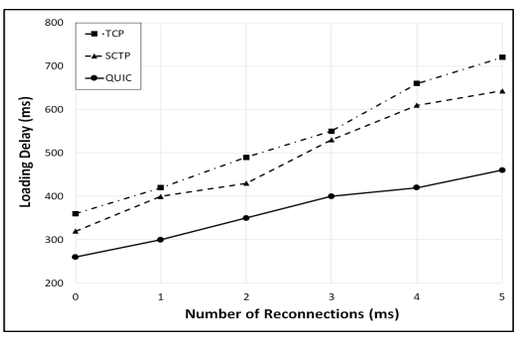

Fig. 13. Loading Delay for Different Number of Reconnections

가진 파일을 다른 스트림을 통해 전달하기 때문에 평균 링크 지연이 높은 환경에서는 SCTP 및 QUIC 방식의 좋은 성능을 보여주었다. 전체적으로 QUIC 방식이 가장 좋은 성능을 보여주고 있다.

Fig. 12와 13은 각각 다양한 패킷오류율 및 재연결 횟수에 따른 Loading Delay를 비교하고 있다. 일반적으로 이동통신망 환경은 이더넷 또는 WiFi 환경에 비해 오류율이 높고 재연결 횟수가 많다고 볼 수 있다. 실험을 통해 우리는 이동통신망 환경에서 더 좋은 QoE를 제공하는 프로토콜을 파악해 볼 수 있다.

Fig. 12에서 QUIC 방식은 모든 오류율 환경에서 가장 좋은 성능을 보여주고 있으며 오류율이 높을수록 프로토콜 간 성능 차이는 더욱 커지고 있음을 확인할 수 있다. 이를 통해 우리는 QUIC 프로토콜이 TCP 및 SCTP에 비하여 전송성능뿐만 아니라 QoE 측면에서도 우수한 기법임을 알 수 있다.

Fig. 13은 재연결 횟수가 증가하는 경우에 대한 QUIC 프로토콜의 효율성을 보여주고 있다. QUIC은 IP주소 변경으로 인하여 재연결을 수행해야 하는 경우에도, TCP 및 SCTP와는 달리 0-RTT 기능을 활용하여 전송 성능 및 높은 QoE를 유지할 수 있다. 이러한 특징으로 인하여 재연결이 많은 이동통신 환경에서도 QUIC은 매우 좋은 QoE를 제공할 수 있음을 확인할 수 있다.

5. 결 론

이 논문에서는 최근 IETF에서 표준으로 개발되고 있는 QUIC 프로토콜에서 대한 성능분석을 수행하였다. 사물인터넷 등의 다양한 인터넷 서비스가 등장함에 따라, 기존 수송계층 프로토콜의 한계를 극복하고 보다 높은 품질의 인터넷 서비스 제공을 위해 QUIC 프로토콜이 개발되었다. QUIC은 TCP의 복잡한 연결설정 과정 및 HoLB 문제 등을 극복하기 위해 제안되었고, 0/1-RTT 및 멀티스트리밍 등의 다양한 특징을 제공한다.

이 논문에서는 HTTP 기반 웹 서비스 및 멀티미디어 스트리밍 서비스를 대상으로 기존 TCP 및 SCTP 프로토콜과 QUIC 프로토콜의 성능을 비교 분석하였다. 개발 언어, 프레임워크 등에 의해 발생할 수 있는 성능 차이를 줄이기 위해 우리는 각 프로토콜의 Go 언어 기반 대표 라이브러리만 활용하여 간단한 테스트베드를 구현하여 성능비교를 수행하였다. 비교를 통해 우리는 평균 링크지연, 패킷오류율 및 재연결 횟수가 높은 환경에서 QUIC 프로토콜이 다른 프로토콜에 비해 더 좋은 성능을 보여준다는 것을 확인할 수 있었다.

향후 연구 이슈로서 QUIC 프로토콜의 기본적인 특성에 대한 성능검증 뿐만 아니라, connection migration 등의 다양한 특징들에 대한 성능 비교분석에 대한 연구도 수행될 필요가 있다.

References

[1] Behrouz A. Forouzan, “TCP IP/Protocol Suite,” McGraw-Hill Inc, 2002.

[2] R. Stewart and C. Metz, “SCTP: new transport protocol for TCP/IP,” *IEEE Internet Computing*, Vol.5, No.6, pp.64-69, Nov, 2001.

[3] J. Posetel, “User Datagram Protocol,” STD 6, RFC 768, IETF, 1980.

[4] J. W. Shon and S. M. Hong, "Provision of Responsive Web User Interface Service for Wireless Router Network Setting in IoT Home Hubs and Devices," *Journal of the Korea Institute of Construction Safety*, Vol.45, No.2, pp.368-374, 2020.
[5] B. H. Andreas, A. M. Tim, and G. Tor-Morten, "Progressive Web Apps: The Possible Web-native Unifier for Mobile Development," *International Conference on Web Information Systems and Technologies*, Vol.2, pp.344-351, 2017.
[6] C. Quentin and B. Olivier, "Multipath QUIC: Design and Evaluation," *Proceedings of the 13th International Conference on Emerging Networking Experiments and Technologies*, pp.160-166, 2017.
[7] R. Minakshi, A. Shamsh, K. Gaurav, and V. Ajay, "Implementation of Quick UDP Internet Connections," *International Journal of Engineering and Computer Science*, Vol.9, No.1, pp.24921-24924, 2020.
[8] C. Gaetano, C. D. Luca, and M. Saverio, "HTTP over UDP: an experimental investigtion of QUIC," *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pp. 609-614, 2015.
[9] A. T. Saif, "Performance analysis of Google's Quick UDP Internet Connection Protocol under Software Simulator," *Journal of Physics: Conference Series*, Vol.1591, No.1, pp.12-26, 2020.
[10] QUIC Working Group [Internet], https://quicwg.org.
[11] S. Floyd, "A report on recent development in TCP congestion control," *IEEE Communications Magazine*, Vol.39, No.4, pp.84-90, Apr. 2001.
[12] M. Matthew and M. Jamshid, "Forward acknowledgement: refining TCP congestion control," *ACM SIGCOMM Computer Communication Review*, Vol.25, No.4, pp.281-291, 1996.
[13] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky, "An Extension to the Selective Acknowledgement (SACK) Option for TCP," *RFC 2883*, IETF, 2000.
[14] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport draft-ietf-quic-transport-29," IETF, 2020.
[15] K. Hugo, P. Kenneth, and W. Hoeteck, "On the Security of the TLS Protocol: A Systematic Analysis," *Annual Cryptology Conference*, pp.429-448, 2013.
[16] E. Rescorla and T. Dierks, "The Transport Layer Security (TLS) Protocol Version 1.3," *RFC 8446*, 2018.
[17] The Go Programming Language [Internet], https://golang.org.
[18] Rust Programming Language [Internet], https://www.rust-lang.org.
[19] Welcome to Python.org [Internet], https://www.python.org.
[20] Caddy - The Ultimate Server with Automatic HTTPS [Internet], https://caddyserver.com.
[21] S. Cook, B. Mathieu, P. Truong, and I. Hamchaoui, "QUIC: Better for what and for whom?," in *Proceedings of IEEE International Conference on Communications (ICC)*, pp.1-6, 2017.
[22] lucas-clemente/quic-go: A QUIC implementation in pure go [Internet], https://github.com/ishidawataru/sctp.
[23] ishidawataru/sctp: SCTP library for the Go programming language [Internet], https://github.com/lucas-clemente/quic-go.

남 혜 빈
https://orcid.org/0000-0003-0943-3024
e-mail : hb1212@knu.ac.kr
2019년 경북대학교 지구시스템과학부 해양학전공/SW부전공(학사)
2021년 경북대학교 컴퓨터학부(석사)
2021년 ~ 현 재 경북대학교 컴퓨터학부 박사과정
관심분야 : 사물인터넷, 프로토콜, QUIC, 웹서비스

정 중 화
https://orcid.org/0000-0002-5828-9551
e-mail : godopu16@gmail.com
2016년 경북대학교 컴퓨터학부(학사)
2018년 경북대학교 컴퓨터학부(석사)
2018년 ~ 현 재 경북대학교 컴퓨터학부 박사과정
관심분야 : 사물인터넷, 프로토콜, 미래인터넷, 엣지컴퓨팅

최 동 규
https://orcid.org/0000-0001-5437-8656
e-mail : supergint@gmail.com
2015년 경북대학교 컴퓨터학부(학사)
2017년 경북대학교 컴퓨터학부(석사)
2017년 ~ 현 재 경북대학교 컴퓨터학부 박사과정
관심분야 : 사물인터넷, QUIC, 엣지컴퓨팅

고 석 주
https://orcid.org/0000-0003-3429-2040
e-mail : sjkoh@knu.ac.kr
1992년 KAIST 경영과학과(공학사)
1994년 KAIST 경영과학과(공학석사)
1998년 KAIST 산업공학과(공학박사)
1998년 ~ 2004년 ETRI 표준연구센터 선임연구원
2004년 ~ 현 재 경북대학교 컴퓨터학부 교수
관심분야 : 사물인터넷, 멀티캐스트, SCTP, QUIC, 가시광통신